feature vector **114** in a database **116**. The database **116** may be internal or external to the computing device **102**. For example, the database **116** may be remote from the computing device **102** and accessible to the computing device **102** via the Internet.

[0020] The computing device **102** can repeat the above process for any number and combination of software projects to automatically build (e.g., construct) the database **116**. This can yield a database **116** with N entries, which can be hundreds or thousands of entries for hundreds or thousands of software projects. Each entry can include a relationship between a software project and its feature vector. For example, the database **116** depicted in FIG. **1** includes a relationship between Software Project A and Feature Vector A, another relationship between Software Project B and Feature Vector B, and another relationship between Software Project N and Feature Vector N.

[0021] These entries in the database **116** may be searchable and comparable to perform various computing tasks. For example, the computing device **102** can receive one or more search queries **118** from a client device **120**, which may be a desktop computer, laptop computer, or mobile device. A search query **118** can be a request to identify one or more software projects in the database **116** that have a particular set of software features. The computing device **102** can then execute a search process in response to the one or more search queries **118**.

[0022] In some examples, the search process can involve the computing device **102** generating a feature mask **124** based on the particular set of software features associated with a search query **118**. A feature mask **124** is a data structure containing elements with values defining criteria for a search. The data structure may or may not be a vector. The feature mask **124** may have as many elements as are present in the feature vectors, and the feature mask **124** may have a similar mapping of elements-to-software features as the feature vectors. In some examples, each element in the feature mask **124** can be a binary value indicating whether or not a software feature corresponding to the element is to limit the search. A binary value of **1** may indicate that presence of the software feature is required, while a binary value of **0** may indicate that the presence of the software feature is optional.

[0023] For example, the computing device **102** can determine the feature mask **124** associated with the search query **118** by first generating a default feature mask in which all the elements have default values (e.g., zeros). One simplified example of the default feature mask can be {0, 0, 0, 0, 0, 0}. Each element in the default feature mask can be mapped to a particular software feature. For example, the elements in the above default feature mask can be mapped to the following: {C++, Openshift, Tensorflow, Linux, Machine-learning, Python}. If the search query **118** indicates that a particular software feature is to be searched, the computing device **102** can then modify the corresponding element's value in the default feature mask to so indicate. For example, if the search query **118** is for software projects that are compatible with Openshift and Linux, then the computing device **102** can change the second and fourth elements' values to one, yielding a feature mask **124** of {0, 1, 0, 1, 0, 0}.

[0024] After determining the feature mask **124**, the computing device **102** can apply the feature mask **124** to some or all of the feature vectors in the database **116** to determine search results, which may consist of a subset of feature vectors having the particular set of software features requested in the search query **118**. Applying the feature mask **124** to a feature vector in the database **116** may involve comparing the feature mask **124** to the feature vector or performing mathematical operations using the feature vector and the feature mask **124**. For example, applying the feature mask **124** to the feature vector in the database **116** may involve comparing the feature mask **124** to the feature vector determine if the feature vector has values of one at the appropriate elements designated in the feature mask **124**. As another example, applying the feature mask **124** to the feature vector in the database **116** may involve performing a bitwise AND operation between the feature mask **124** and the feature vector. As yet another example, applying the feature mask **124** to the feature vector in the database **116** may involve determining a distance (e.g., Euclidian distance) between the feature mask **124** and the feature vector, where the computing device **102** may return as search results only those feature vectors having distances that are below a predefined threshold. Regardless of the approach, the computing device **102** determine one or more software projects having the particular set of features identified in the search query **118** by using the feature mask **124**.

[0025] In some examples, the search query **118** may be a request to identify a closest match to a software project **122**. The closest match can be another software project having the largest number of software features in common with the software project **122** among all of the other software projects in the database **116**. In some examples, the computing device **102** can determine the closest match by determining distances (e.g., Euclidian distances) between the feature vector for the software project **122** and each of the other feature vectors for the other software projects in the database **116**. Whichever of the other software projects in the database **116** has the closest distance to the feature vector for the software project **122** can be the closest match to the software project **122**.

[0026] In some examples, the search query **118** may be for some combination of the above. As one particular example, the software project **122** may be Tensorflow and have a feature vector of {1, 1, 0, 0, 0, 0, 0, 1, 0}. The feature vector may correspond to the following software features, respectively: {python, machine-learning, web, Django-framework, webassembly, sql, spark, gpu-support, java}. Thus, the feature vector indicates that Tensorflow requires Python. The user submit a search query **118** for other software projects in the database **116** that are similar to Tensorflow, but suitable for a Java ecosystem rather than a Python ecosystem. To effectuate such a search, the computing device **102** can generate a feature mask **124** based on the search query **118**. One example of the feature mask **124** is {0, 1, 1, 1, 1, 1, 1, 1, 1}. The computing device **102** can then apply this feature mask **124** to all of the feature vectors in the database **116** to determine a refined search space that excludes software projects requiring Python. The computing device **102** can also generate a modified feature vector for Tensorflow based on the feature mask **124**. For example, the computing device **102** can apply a bitwise AND operation between the feature vector for Tensorflow and the feature mask **124** to determine a modified feature vector of {0, 1, 0, 0, 0, 0, 0, 1, 0}. The computing device **102** can then determine distances between the modified feature vector for Tensorflow and each of the other feature vectors for the other software projects in